

DATA MANAGEMENT APPLIANCE**1. Field of the Invention:**

5 The present invention is directed generally toward data storage and backup systems. More specifically, the present invention is directed toward a backup system that immediately backs up data written to a primary storage device to multiple "virtual mirrors," that reflect the
10 changing state of the primary storage device over time.

2. Background of the Invention:

Humankind has always had a need to record information. Historians tell us that in ancient
15 Mesopotamia, writing first emerged as a means of keeping records of livestock. As civilization progresses, so does the need to securely store larger amounts of information for longer periods of time. Whereas in ancient times, clay tablets sufficed for most storage
20 needs, modern computerized storage systems are measured in such seemingly astronomical terms as gigabytes and terabytes. One example of this information storage explosion is the U.S. Internal Revenue Service's use of computers to store information regarding taxable gifts
25 made over a person's lifetime. For most people living in the United States, gift taxes are not calculated or paid until death, so any information regarding taxable gifts must be maintained over a person's lifetime.

Although computerized storage is somewhat more
30 robust than brittle clay tablets, the problem of

Docket No. 2001-054-SFT

maintaining reliable storage over a long period of time remains. For this reason, many, if not most, large-scale computing facilities periodically back up stored data to some redundant storage medium, such as to tapes. There are two types of backups that are generally performed in computer systems today. Full backup means making a redundant copy of a storage system in its entirety. Incremental backup, on the other hand, means making a redundant copy of only those portions of a storage system that have changed since the last backup. Many computing facilities make use of both full backup and incremental backup.

A number of problems exist with these backup methods, however. Firstly, a "backup window" of time must usually be available when computer applications are shut down so that a consistent image of the storage system can be made (i.e., so that what is being copied does not get overwritten by an application while the copy is being made). Second, even if no backup window is necessary, the backup process, when run as a batch operation, can steal CPU cycles from other processes running on the computer system. Third, so-called primary storage devices, such as disk, are today very large, so that backing up data sequentially to a secondary storage medium such as tape and recovering data from the tape are relatively slow operations. Fourth, since most backup systems today operate at the file-system level, backup systems must contend with complex directory-structure and security issues. Fifth, with backups being performed only periodically, there is a high risk of data loss,

Docket No. 2001-054-SFT

because data written between backups may be lost between backups. Sixth, existing replication solutions tend to be expensive. Seventh, costs associated with media and device incompatibilities are high.

5 In the database design field, recovery without a backup window is often accomplished through the use of write-ahead logging. Database transactions that can change database contents are recorded in a log before being completed in the main database. Another name for a
10 log is "journal." If the database becomes corrupted, transactions can be "undone" or "redone" to restore the database to some previous uncorrupted state.

Another recovery technique used in the database field is "shadow paging." Shadow paging divides database
15 contents into a series of pages. A directory is used to map logical addresses for pages into physical addresses on a storage device. When changes are made to the database, the pages are not overwritten, but new pages containing the changes are produced, and a new directory
20 is created that points to the new pages instead. Recovery is performed by reverting to a directory from a previous, uncorrupted state in the database.

U.S. Patent No. 5,086,502 to Malcolm extends the write-ahead logging concept to primitive disk I/O.
25 Malcolm describes a system wherein write commands to a storage device in an IBM PC-type computer system are captured at the BIOS (basic input/output system) level and recorded in a journal. Write commands recorded in the journal are then used to restore the storage device
30 to an earlier, uncorrupted state.

Docket No. 2001-054-SFT

U.S. Patent 6,158,019 to Squibb describes a method and apparatus for restoring an updated computer storage system from a journal of write events. Squibb describes process whereby events in an event journal may be used to
5 create an event map and "delta" data structure, which may be merged with an original file stored on streaming media to generate a previous version of a file.

Both of these data replication strategies, however, involve elaborate steps of data reconstruction and use a
10 disproportionately large amount of storage space over time. Thus, they can be unwieldy and expensive to maintain and use. Additionally, the Squibb and Malcolm systems place a heavy computational burden on the primary (host) computer system. What is needed is a data
15 replication system that eliminates the backup window, is fast, and makes more efficient use of storage space, without placing a heavy computational burden on the primary or host computer.

SUMMARY OF THE INVENTION

The present invention is directed toward a data management appliance and ancillary technologies for replicating data written to a primary storage system. The data management appliance is a random-access storage system that at the logical block level replicates the contents of a primary storage system over time. A mirror-in-the-middle (MIM) included in the data management appliance is used to record an exact copy of the primary storage system at some fixed point in time.

Atomic write events are recorded in a "forward journal" by the appliance immediately, so that applications are not interrupted. An atomic event or transaction is one that cannot be divided into parts; an atomic event or transaction is either performed in its entirety or not performed at all. At specified points in time, forward journal entries are used to produce snapshots, reflecting the change in the primary storage system over a period of time. These snapshots are recorded in a "backward journal" and represent a coarser level of backup granularity, much like an incremental backup. As snapshots are produced, the forward journal entries may be applied to the MIM to update its contents and finally discarded to save space.

A virtual recovery mapping object (VRMO) is used to map logical addresses at a particular point in time to their physical locations within the data management appliance. Thus, VRMOs act as an index, allowing for quick, random-access recovery of data. In one

Docket No. 2001-054-SFT

embodiment, a VRMO is composed of a multi-way tree, which allows logical address translation in logarithmic time.

As the data management appliance allows the contents of a storage system over a period of time to be examined, 5 the data management appliance may be applied to the detection and/or forensic investigation of data events, such as a database corruption or viral infection.

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
220

BRIEF DESCRIPTION OF THE DRAWINGS

The novel features believed characteristic of the invention are set forth in the appended claims. The invention itself, however, as well as a preferred mode of use, further objectives and advantages thereof, will best be understood by reference to the following detailed description of an illustrative embodiment when read in conjunction with the accompanying drawings, wherein:

10 **Figure 1** is an overall view of the operation of a data management appliance in accordance with a preferred embodiment of the present invention;

15 **Figure 2** is an overall view of the operation of an alternative embodiment of the present invention in which commands are replicated using a replicating controller;

Figure 3 is a diagram providing a conceptual view of the replicated storage provided by a data management appliance in accordance with a preferred embodiment of the present invention;

20 **Figure 4** depicts a process of replicating data within a data management appliance in accordance with a preferred embodiment of the present invention;

25 **Figure 5** depicts the basic operation of a virtual recovery mapping object (VRMO) in accordance with a preferred embodiment of the present invention;

Figure 6 is a diagram depicting two VRMO types usable in a preferred embodiment of the present invention;

Figure 7 is a diagram of a forward journal in accordance with a preferred embodiment of present invention;

Figure 8 is a diagram of a mirror-in-the-middle (MIM) in accordance with a preferred embodiment of the present invention;

Figure 9 is a diagram representing an overall view of a process of updating a MIM and generating snapshots in accordance with the preferred embodiment of the present invention;

Figure 10 is a flowchart representation of a process of generating a new snapshot and bringing a MIM into synchronization with primary storage in accordance with a preferred embodiment of the present invention;

Figure 11 is a diagram depicting a process of generating a VBMM in accordance with a preferred embodiment of the present invention

Figure 12 depicts a process of generating a backward movelist from a PEL (physical extent list) in accordance with a preferred embodiment of the present invention;

Figure 13 is a diagram depicting incorporation of a backward movelist into a backward journal in accordance with a preferred embodiment of the present invention;

Figure 14 is a diagram demonstrating insertion of moves from a backward movelist into a new VBMM in accordance with a preferred embodiment of the present invention;

Figure 15 is a flowchart representation of a process of generating a snapshot, including a VBMM and backward

Docket No. 2001-054-SFT

journal entries, according to a preferred embodiment of the present invention;

Figure 16 is a diagram depicting a process of updating an older VBMM with a backward movelist in accordance with a preferred embodiment of the present invention;

Figure 17 is a diagram that provides an example of a process of updating a VBMJ in accordance with a preferred embodiment of the present invention;

Figure 18 is a flowchart representation of a process of updating a VBMJ and generating backward journal entries, according to a preferred embodiment of the present invention;

Figure 19 depicts an exemplar multi-way VBMJ tree data structure for representing a mapping from logical storage device addresses to physical journal/snapshot addresses in accordance with a preferred embodiment of the present invention;

Figure 20 is a flowchart representation of a process of generating storage replicas in accordance with a preferred embodiment of the present invention;

Figure 21 is a diagram that depicts a process of monitoring a database for violation of consistency constraints in accordance with a preferred embodiment of the present invention;

Figure 22 depicts a system that monitors for viruses in accordance with a preferred embodiment of the present invention;

Figure 23 is a flowchart representation of a process of monitoring for troublesome changes in data backed up

Docket No. 2001-054-SFT

by a data management appliance in accordance with a preferred embodiment of the present invention;

Figure 24 is a diagram that depicts a single data management appliance attached to a storage network with multiple servers having attached primary storage devices being attached to a storage network in accordance with a preferred embodiment of the present invention;

Figure 25 is a diagram depicting a single computer system console controlling a single primary storage device that is being backed up by multiple data management appliances working in tandem through a storage area network in accordance with a preferred embodiment of the present invention; and

Figure 26 is a diagram depicting a data processing system wherein the data management appliances share common pooled storage in accordance with a preferred embodiment of the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

Figure 1 is a diagram providing an overall view of the operation of a preferred embodiment of the present invention. A primary storage application **100**, residing on a host computer system, submits a command to a replication driver/volume manager **102** to store data to primary storage **108**. Replication driver/volume manager **102** relays the request both to disk driver **104** and network driver **110**. Disk driver **104** is device driver code that operates disk controller **106**, which in turn controls primary storage **108**, which is in this case a disk drive, although many different alternative random-access storage devices could be used in place of primary storage **108**.

Network driver **110** is device driver code that controls access to a computer network. Network driver **110** relays the submitted storage command to data management appliance **112**, which is located on a network associated with the host computer system. Data management appliance **112** is an intelligent peripheral device that presents the appearance on the network of a disk array or arrays or other random-access storage medium. Data management appliance **112** contains control circuitry and also contains its own random-access storage **114**. The control circuitry may be, for instance, an embedded stored-program computer, such as a microprocessor and memory or a microcontroller. The stored program may be stored in firmware or loaded from a storage medium, such as floppy disk. Data management

appliance 112, through control circuitry, uses random-access storage 114 to replicate the information stored on primary storage 108. As will be seen, data management appliance 112 not only provides a replica of the current
5 contents of primary storage 108, but it also contains information that it can use to reconstruct replicas of previous contents of primary storage 108 at various points in time.

Figure 2 is a diagram of an alternative embodiment of the present invention, in which the replication driver is replaced with replicating hardware. Primary storage application 200 issues a write command to disk driver 202, which is device driver code. Disk driver 202 controls replicating controller 204, which is a hardware
15 disk controller that controls primary storage 206, but which has the additional feature of replicating storage commands submitted to primary storage 206 and providing the replicated commands to data management appliance 208, which replicates the data contained on primary storage
20 206 on random-access storage 210.

Figure 3 is a diagram providing a conceptual view of the replicated storage provided by data management appliance 112 (or 208). Data management appliance 112 is programmed to behave as though it stores a number of
25 duplicate copies (replicas) of primary storage device 108 as it existed at certain points in time (replicas 300, 302). Data management appliance 112 can provide a near-time (i.e., near the present time) replica (300) of primary storage device 108 or any one of a number of
30 virtual views or mirrors of earlier versions of the data

Docket No. 2001-054-SFT

stored on primary storage device 108. Each of these virtual mirrors is accessed using one of a number of virtual recovery mapping objects (VRMOs) 304, which each represent a different point in time.

5 Data may be read from data management appliance 112 by either specifying that data management appliance 112 behave (for reading purposes) as a copy of primary storage device 108 at a specified time (e.g., during the mounting process), or by specifying read commands that
10 contain an additional time field. For example, to retrieve the contents of block 5 at some time "t," either data management appliance 112 could be directed to behave as if it were time "t," in which case any read command to any block would result in the data that was present at
15 time "t," or a read command that simply stated "retrieve block 5 from time 't'" could be issued instead.

Figure 4 depicts a process of replicating data within a data management appliance in accordance with a preferred embodiment of the present invention. The
20 replication process centers around "mirror in the middle" (MIM) 400, which initially stores an identical copy of the data stored on the primary storage device (108 in **Figure 1**). MIM 400 is a reserved portion of random-access storage 114, which is identical in capacity and
25 address space as primary storage 108. After an identical copy of primary storage 108 has been established on MIM 400, subsequent write commands issued to change the data contents of primary storage device 108 are archived sequentially in forward journal 402, without changing the
30 data stored in MIM 400. Thus, forward journal 402

Docket No. 2001-054-SFT

contains the entire sequence of write commands issued since MIM 400 was identical with primary storage device 108. Forward journal 402 and MIM 400 are both stored in random-access storage 114.

5 After the passage of a certain period of time, either a pre-defined time interval or when the portion of random-access storage 114 devoted to forward journal 402 is exhausted of space, a portion of the archived commands in forward journal 402 consisting of the oldest command
10 in the journal and some number of commands following the oldest command in sequence, up to a point in time determined by the archiving policy are combined so as to obtain a net change taking place during the time period. For example, if storage addresses 1-5 are written to with
15 the string "ABCDE," then addresses 2-4 are rewritten with "ZXC," the net change to the data is to write "AZXCE" to addresses 1-5. As shown in **Figure 13**, these changes can be expressed in terms of a "backward movelist move" including a logical address within the logical address
20 space of primary storage 108 where the change occurs, a length of the change, and an address of a location within the forward journal at which the newly-written data is stored.

 Next, the starting addresses and lengths
25 representing the net change are used to retrieve data from MIM 400 to derive an inverse of the net change. For example, if addresses 1-5 originally contained "12345," and the net change is to write "AZXCE" to addresses 1-5, then the inverse of the net change is to write the
30 original "12345" to addresses 1-5, which reverses the net

Docket No. 2001-054-SFT

change made. This inverse net change is then recorded as a "snapshot" in backward journal 404 and MIM 400 is updated to reflect the determined net change.

Thus, recent copies of primary storage device 104 may be retrieved by applying subsequent changes from journal 402 to the data stored in MIM 400, while more distant copies may be retrieved by applying the reverse changes (snapshots) to MIM 400. Since primary storage device 108 is journaled with finer granularity for more recent transactions than for more distant transactions, a balance is struck between the ability to restore data at an exact moment in time and the ability to save space by storing a sparse number of snapshots of the data.

The system described in **Figure 4** can be further enhanced by allowing for the archiving of past-time data onto removable media, such as image tape 406 and difference tape 408. An image tape, such as image tape 406, containing a complete copy of primary storage device 104 at a particular time, can be assembled from MIM 400 and snapshots 404. A difference tape, such as difference tape 408, archiving the net differences between an image tape and successive snapshots, can also be generated. Because they are stored on removable media, image tape 406 and difference tape 408 can be stored away (e.g., in a tape silo or library unit) for future use without tying up system resources, such as tape or disk drives.

Figure 5 depicts the basic operation of a virtual recovery mapping object (VRMO) in accordance with a preferred embodiment of the present invention. As was shown in **Figure 4**, portions of replicated past data may

Docket No. 2001-054-SFT

be stored in the journal or in snapshots. Thus, a "virtual replica" of the primary storage device will generally be composed of various fragments stored across snapshots. To access the virtual mirror, one must
5 identify where each piece of data is stored within the various snapshots in the backward journal, in the forward journal, or on the MIM.

As **Figure 5** shows, VRMO 502, which is associated with a particular time "t," is a data structure that maps
10 a logical address (500), representing the location on primary storage device 108 of the data to be accessed, to a physical address (504), representing the position of the data within a snapshot or journal. VRMO 502 will preferably embody an index for rapid lookup of physical
15 address 504 given logical address 500. VRMO 502 is preferably stored in memory contained within the control circuitry of data management appliance 112.

Figure 6 is a diagram depicting two VRMO types usable in a preferred embodiment of the present
20 invention, VBMM (Virtual Block Map - MIM) 600 and VBMJ (Virtual Block Map - Journal) 602. VBMM 600 and VBMJ 602 are depicted here as binary search trees, although a multi-way tree structure such as that depicted in **Figure 19** could be used as well. Search trees, and in
25 particular binary search trees, are a common data structure for indexing data having an ordering characteristic and are well known in the computer programming art. VBMM 600 and VBMJ 602 are both marked with timestamps (604 and 606) denoting the points in time
30 represented by VBMM 600 and VBMJ 602.

Docket No. 2001-054-SFT

VBMM 600 is "MIM-centric." This means that VBMM 600 is indexed with respect to blocks of data contained on MIM 400. Tree nodes 608 of VBMM 600 represent contiguous regions of the logical address space that are contained
5 on MIM 400. Leaf nodes 609 point to physical extent lists (PELs) 610, which represent portions of the logical address space, adjacent to the leaf-node addresses, that are stored in journal entries (snapshots from the backward journal or segments from the forward journal).

10 In VBMM 600, logical addresses are resolved into physical addresses by first traversing tree nodes 608 to attempt to find the logical address contained on MIM 400. If the logical address being sought is contained in an address range associated with a tree node, the logical
15 address simply resolves into an identical physical address on MIM 400, since the data being sought is located on MIM 400. If, on the other hand, the address being sought is not contained within a tree node, the proper physical extent list adjacent to the last tree
20 node searched will be searched for the physical address within the journal that corresponds to the logical address being sought. It should be noted that physical extent lists 610 are pointed to by leaf nodes 609 as left and right children, just as if each of the physical
25 extent lists were inserted as tree nodes within the tree structure. For example, a physical extent list containing addresses that are less than that of its parent tree node will be the left child of that parent tree node.

VBMJ (Virtual Block Map - Journal) 602 is, in contrast, journal-centric. Tree nodes 612 represent ranges of logical addresses contained within journal entries. Tree nodes 612 are mapped in one-to-one
5 correspondence with physical extent lists 614, which, like those of VBMM 600, map logical addresses into physical addresses located within journal entries.

Resolving a logical address into a physical address using VBMJ 602 is straightforward. If the address in
10 question resides within the journal, traversing the tree of VBMJ 602 will result in finding a tree node and corresponding physical extent list mapping the logical address into its physical location in the journal. If the address resides on the MIM instead, the tree search
15 will fail. In that case, the proper physical address is on the MIM and is equivalent to the logical address.

Figure 7 is a diagram of a forward journal 700 in accordance with a preferred embodiment of present invention. Forward journal 700 is made up of two queues,
20 701 and 703. Queues 701 and 703 are, in a preferred embodiment, circular queues, to minimize the amount of persistent directory structures needed to decipher the state of the queue after an interruption in power. A circular queue only needs to persist pointers to the head
25 and tail of the queue to be fully recoverable. Queues, and in particular circular queues, are well known data structures within the computer programming art.

The elements of queue 701 are composed primarily of extent descriptors 702. Each of extent descriptors 702
30 describes a contiguous extent of data within the logical

Docket No. 2001-054-SFT

address space. Each of extent descriptors **702** includes a logical start address **704**, a length **706** and a physical address **708**. Logical start address **704** represents the starting address of the extent of data within the logical address space. Length **706** represents the length of the extent of data. Physical address **708** represents a physical address within queue **703** in which the extent is stored. Queue **701** may include one or more sync indicators, such as sync indicator **710**, which act as time markers.

Queue **703** is made up of extents **712**. Extents **712** represent the actual data written to primary storage **108** and replicated by data management appliance **112** in forward journal **402** (see **Figures 1** and **4**).

Figure 8 is a diagram of a MIM **800** in accordance with a preferred embodiment of the present invention. MIM **800** is divided into storage units or blocks such as block **804**. The storage units or blocks can be any fixed-size data aggregation and depend on the organization of the primary storage. Block **804**, like the other storage units in MIM **800**, has a logical address **802** associated with it. These logical addresses are the same logical addresses within the aforementioned logical address space. Thus, MIM **800** represents the entire logical address space. The same logical address space is used to represent storage locations within the primary storage.

Figure 9 is a diagram representing an overall view of a process of updating a MIM and generating snapshots in accordance with the preferred embodiment of the present invention. As time progresses and data is

Docket No. 2001-054-SFT

written into primary storage 108 and forward journal 402, a point will be reached at which a snapshot is produced. Preferably, the appliance selects a point at which a sync indicator such as sync indicator 710 occurs in the
5 forward journal to be the endpoint of the snapshot.

VRMO 900 maps addresses from the logical address space to physical locations on the MIM (400 in **Figure 4**) or the backward journal (404 in **Figure 4**). At the very beginning of the data replication process, MIM 400 and
10 primary storage 108 are identical. Thus initially, VRMO 900 maps logical addresses into physical locations on MIM 400 only.

When an update event occurs, outstanding forward journal entries 902 are used to modify VRMO 900 to
15 produce a new VRMO 906. Meanwhile, VRMO 900 and forward journal entries 902 are used to produce new backward journal entries 908. Backward journal entries 908 and VRMO 906 define a snapshot 910. Finally, MIM contents 904 are updated using forward journal entries 902 to
20 produced new MIM contents 912, thus bringing MIM 400 forward in time, particularly, so as to represent the point in time indicated by sync indicator 710.

Figure 10 is a flowchart representation of a process of generating a new snapshot and bringing a MIM forward
25 in time in accordance with a preferred embodiment of the present invention. First, a forward journal move list is generated from the forward journal (step 1000). This forward journal move list contains the extents of all outstanding journal entries from queue 701 in **Figure 7**.
30 Next, a backward journal move list is generated from the

Docket No. 2001-054-SFT

forward journal move list in current VRMO (step 1002). In addition, a new VRMO is also generated (step 1002). The backward journal move list contains extents, just as the forward journal move list. Unlike the forward
5 journal move list, however, the backward journal move list represents locations on MIM 400 that will be overwritten when MIM 400 is brought forward in time. These locations must be preserved in backward journal to allow previous versions of the contents of primary
10 storage 108 to be recovered, particularly, to allow the point in time represented by the MIM immediately before the MIM update event to continue to be recoverable. Thus, these locations are copied from the MIM into the backward journal according to the backward journal move
15 list (step 1004). Next, MIM 400 is updated to reflect each of the write commands contained in the forward journal move list (step 1006). Finally, the portion of the forward journal that was used to update the MIM is now relieved or recycled to make room for new incoming
20 journal transactions (step 1008).

Figure 11 is a diagram depicting a process of generating a VBMM in accordance with a preferred embodiment of the present invention. Initially the VBMM consists of a single node 1100 representing the address
25 range of the entire logical address space. The VBMM is constructed by applying a recursive algorithm to divide node 1100 into a tree of nodes covering smaller ranges. The recursive algorithm takes as its input a single address range representing a move from the forward move
30 list and a current node.

At each node in the current VBMM, the range of the node is compared with the input range. Depending on how or if the ranges overlap, the current VBMM node may reduced in size, deleted, or split. The algorithm may then call itself recursively to traverse the left and/or right children of each node until the leaves of the tree are reached. Also, a physical extent list (PEL) is generated, which will become the backward journal movelist. PEL entries will be generated at each node where the input address range overlaps with the node address range. **Table I**, below, is a decision table showing the proper actions associated with each degree of overlap between the input range and the node range. The abbreviations MLS and MLE refer to the starting and ending addresses of the range represented by the input move from the movelist and VBS and VBE refer to the starting and ending addresses of the range represented by the current node.

Table I: Decision Table for VBMM Update

Decision Points		Outcomes			
Start fact	End fact	Left recursion	Right recursion	VBMM node modification	Emit physical extent list entry
Don't Care	MLE < VBS	[MLS, MLE]	None	None	None
MLS > VBE	Don't Care	None	[MLS, MLE]	None	None
MLS = VBS	MLE = VBE	None	None	Delete	[MLS, MLE]
MLS = VBS	MLE > VBE	None	[VBE+1, MLE]	Delete	[MLS, VBE]
MLS = VBS	MLE < VBE	None	None	Shrink [MLE+1, VBE]	[MLS, MLE]
MLS > VBS	MLE = VBE	None	None	Shrink [VBS, MLS-1]	[MLS, MLE]
MLS > VBS	MLE > VBE	None	[VBE+1, MLE]	Shrink	[MLS, VBE]

Docket No. 2001-054-SFT

VBS	VBE			[VBS,MLS-1]	
MLS > VBS	MLE < VBE	None	None	Split [VBS,MLS-1] [MLE+1,VBE]	[MLS, MLE]
MLS < VBS	MLE = VBE	[MLS,VBS-1]	None	Delete	[VBS, MLE]
MLS < VBS	MLE > VBE	[MLS,VBS-1]	[VBE+1,MLE]	Delete	[VBS, VBE]
MLS < VBS	MLE < VBE	[MLS, VBS-1]	None	Shrink [MLE+1,VBE]	[VBS, MLE]

Turning now to the example in **Figure 11**, a first input move **1102** is taken from the forward movelist. Move **1102** includes a starting logical address **1104**, a length of data to be written **1106**, and a physical address **1108** within the forward journal where the data is stored. As move **1102** represents a write to an address range that is fully contained within the address range of node **1100**, node **1100** is split into two nodes, **1110** and **1111**. A PEL entry **1112** is also generated, containing a logical starting address **1114** and ending address **1116**.

Next, a new move **1118** is submitted to the algorithm. As the range described in move **1118** overlaps the end of the range in node **1110**, node **1110** is reduced in size to generate node **1120** and PEL entry **1124** is generated, according to the appropriate decision table rule from **Table I** above.

The algorithm is then called again (right-recursively) with the adjusted input range of (8:13). As this range overlaps that of node **1111** at the beginning of node **1111**'s range (12:100), node **1111** is shortened in range to produce node **1121** and a new PEL entry **1126** is generated.

Figure 12 depicts a process of generating a backward movelist from a PEL **1200** in accordance with a preferred

Docket No. 2001-054-SFT

embodiment of the present invention. First, PEL 1200 is sorted in ascending order by starting address to achieve PEL 1202. Then, those series of PEL entries representing a contiguous block of logical addresses are combined into
5 a single entry (e.g., 1204).

This newly formed backward movelist can then be incorporated into the backward journal as shown in **Figure 13**. The entries (moves) from the backward movelist (e.g., 1204) are inserted into a first queue 1300. A
10 sync marker 1302 represents the beginning of backward journal entries for the present snapshot. Backward journal entries are written as extent descriptors, such as extent descriptor 1304. Extent descriptor 1304 contains a logical starting address 1306, a length 1308,
15 and a physical address 1310. Starting address 1306 and length 1308 are derived from backward move 1204. Physical address 1310 refers the starting location within a second queue 1312 corresponding to starting address 1306. Data at logical starting address 1306 is copied
20 from the MIM and into second queue 1312 at physical address 1310.

Finally, as shown in **Figure 14**, each of the moves in the backward movelist (e.g., move 1204) are inserted into the new VBMM (1400) as "physical extent nodes" (e.g.,
25 1402), to data stored in the backward journal to be located using VBMM 1400.

Figure 15 is a flowchart representation of a process of generating a snapshot, including a VBMM and backward journal entries, according to a preferred embodiment of
30 the present invention. For each forward journal entry,

Docket No. 2001-054-SFT

the VBMM (which initially contains a single node representing the entire logical address space) is traversed and new VBMM nodes and PEL entries are generated according to the decision table above (step 5 1500). The new PEL is sorted (step 1502). Next, contiguous PEL entries are joined together (step 1504). The new PEL entries are then placed into a backward journal movelist (step 1506). Data corresponding to the backward journal entries is then copied from the MIM into 10 the backward journal (step 1508). Finally, the new PEL entries are inserted into the VBMM tree as physical extent nodes (step 1510).

When all of the journal entries that are to be accounted for in a single snapshot have been placed into 15 the backward journal and VBMM (or VBMJ), we say the snapshot is "closed." In practice, we can perform several synchronization events (where we create backward journal entries and update the MIM) while the snapshot is still "open." Further, even if the snapshot is closed, 20 when data from a particular snapshot gets updated, and thus overwritten on the MIM, the VBMM (or VBMJ) of the older snapshot can no longer rely on the MIM and must be updated to point to a copy of the overwritten data in the backward journal.

25 **Figure 16** depicts a situation in which a VBMM 1600 for a snapshot is modified to generate a new VBMM 1602 for the snapshot when a synchronization event occurs. What happens is this: The algorithm just mentioned for producing a new VBMM is called, just as if the VBMM was 30 new, except for two differences. The first and most

Docket No. 2001-054-SFT

obvious difference is that the old VBMM is traversed, rather than the original single node. The second difference is that the new physical extent nodes (1604), although they may refer to portions of the logical address space that are contiguous with older physical extent nodes, will refer to physical addresses in the backward journal that are not contiguous with those of the older physical extent node. Thus, new physical extent nodes 1604 will have to coexist in a sorted physical extent list (PEL) at the leaves of the tree, but not be combined with older physical extent nodes representing adjacent locations within the logical address space.

Eventually, as more data in a snapshot gets moved from the MIM to the backward journal, it is no longer efficient to traverse the entire VBMM to the leaves to locate data in the journal. In such a case, the VBMM can be converted into a VBMJ. Referring now to **Figure 6**, the process for converting a VBMM (600) to a VBMJ (602) is straightforward. First, all of the PELs (614) in the VBMM are collected. For each PEL, a VBMJ node (e.g., VBMJ nodes 612) is created representing an address range where the starting address of the range is the starting address of the first entry in the PEL and the ending address is the ending address of the last entry of the PEL, and where each VBMJ node points to its respective PEL. Finally, the VBMJ nodes are inserted into the new VBMJ tree.

Just as with VBMMs, VBMJs have an algorithm for updating the VBMJ to point to entries in the backward

journal in the event that that data is overwritten in the
 MIM. An example of this algorithm is provided in **Figure**
17. As with the VBMM algorithm, the VBMJ update
 algorithm involves applying rules from a decision table
 5 at each node encountered during a traversal of tree
 nodes. Unlike the VBMM algorithm, however, three
 additional data structures are needed for the VBMJ
 algorithm.

Since the nodes of a VBMJ represent extents within
 10 the backward journal, as more data becomes copied into
 the backward journal, the nodes of the VBMJ are enlarged
 or merged, unlike the VBMM where nodes are split or
 reduced. It is necessary to store information regarding
 the nodes and physical extents that will be combined in
 15 these three auxiliary data structures. A collapse set C
 is a set of logical address ranges to be combined into a
 single node. A collapse set physical extent set CP is a
 set of physical extent nodes to be included in the
 physical extent list (PEL) for the node generated from
 20 collapse set C. Backward journal movelist candidate set
 BJMC stores a set of logical address ranges to be
 converted into backward journal extents.

As was stated earlier, the algorithm progresses by
 traversing the VBMJ tree, applying decision rules at each
 25 node according to the degree and type of overlap of the
 input move address range ([MLS, MLE]) and the address
 range for the current node ([VBS, VBE]). The decision
 rules for the VBMJ algorithm are listed in **Table II**
 below:

Table II: Decision Table for VBMJ Update

Decision Points		Outcomes
Start fact	End fact	
Don't Care	MLE < VBS-1	Execute decision table for [MLS,MLE] and left child.
Don't Care	MLE = VBS-1	Add current node to C. Execute decision table for [MLS,MLE] and left child.
MLS > VBE+1	Don't Care	Execute decision table for [MLS,MLE] and right child.
MLS = VBE+1	Don't Care	Add current node to C. Execute decision table for [MLS,MLE] and right child.
MLS => VBS	MLE =< VBE	End decision table processing. This movelist entry does not affect this snapshot, since the information from the MIM is already stored in the journal.
MLS => VBS but <= VBE	MLE > VBE	Add current node to C. Execute decision table for [VBE+1,MLE] and right child.
MLS < VBS	MLE <= VBE but => VBS	Add current node to C. Execute decision table for [MLS,VBS-1] and left child.
MLS < VBS	MLE > VBE	Add current node to C. Execute decision table for [MLS,VBS-1] and left child. Execute decision table for [VBE+1,MLE].

When a recursion is ordered, but the child pointer in the indicated direction is NULL (i.e., the tree traversal has reached a leaf node, and the ordered recursion cannot be performed, since the proper child node for further recursion does not exist), then the input range being processed ([MLS, MLE]) is added to C and a corresponding physical extent is added to CP. The current input range is also added to BJMC.

Once the VBMJ has been traversed using the decision rules in **Table II**, the collapse set and affiliated data structures are processed to produce the new VBMJ and backward journal entries. First a new VBMJ node is created but not added to the VBMJ. This node is given an address range that is equivalent to the "range of C", which can be denoted $R(C)$. $R(C)$ has as its starting

Docket No. 2001-054-SFT

address the lowest starting address of the address ranges in C; R(C) has as its ending address the greatest starting address of the address ranges in C (this is because C represents a number of ranges that form a contiguous block of address space).

The PEL of the new VBMJ node is then made to contain all of the physical extents represented in CP, sorted in ascending order by logical starting addresses. Next, all of the VBMJ nodes in the VBMJ corresponding to address ranges contained in C are deleted to make room. Then the new VBMJ node is inserted into the VBMJ to replace the deleted VBMJ nodes.

To complete the synchronization process, new backward journal entries must be created and the MIM updated in accordance with the forward journal entries. This is done by sorting and combining the BJMC set using the process depicted in **Figures 12 and 13**. The new physical extents in the VBMJ that were created to accommodate the new backward journal entries are then updated to point to the physical addresses within the backward journal at which the snapshot data copied to the backward journal from the MIM resides.

Once a VBMJ has been updated, updating VBMJs representing older snapshots is easy. Instead of processing the forward journal movelist, the backward journal movelist is used in its place and the same algorithm applied, with an exception being that the generated physical extent nodes are made to point to data already within the backward journal, rather than recopying the data from the MIM.

Figure 17 is a diagram that provides an example of a process of updating a VBMJ in accordance with a preferred embodiment of the present invention. The example starts with an existing VBMJ 1700 and data structures C 1702, CP 1704, and BJMC 1706. The range of C, R(C) 1708, is also shown for convenience.

A forward journal move 1710 is processed. The logical address range of move 1710, does not overlap at all with node 1711 but is located after the range of node 1711, so node 1720, the right child of node 1711, is examined. Node 1720 overlaps with the range of move 1710. Thus, the range of node 1720 is added to C 1712 and its PEL 1722 is added to CP 1714. As node 1720 is a leaf, the current input range, which is shortened from (42,48) to (46,48) by the proper decision rule from **Table II**, is inserted into C 1712, CP 1714, and BJMC 1716. Thus, the range of C, R(C) 1718, is [39, 48]. Node 1720 is then deleted from the VBMJ and replaced with a new node 1724, whose range is equivalent to R(C) 1718, and whose PEL 1726 includes all of the physical extents in CP 1714.

Figure 18 is a flowchart representation of a process of updating a VBMJ and generating backward journal entries, according to a preferred embodiment of the present invention. First, the forward journal movelist is used to traverse the current VBMJ and generate C, CP, and BJMC according to the decision rules in **Table II** (step 1800). A new VBMJ node is generated with range R(C) (step 1802). The VBMJ nodes contained in C are deleted (step 1804). The new VBMJ node is inserted in

place of the deleted nodes (step 1806). The BJMC set is sorted and contiguous BJMC entries are combined (step 1808). New backward journal entries are generated from the sorted, combined BJMC set (step 1810). Finally, the backward journal entries are used to update any older VBMJs that may exist (step 1812). After all VBMJs and VBMMs are updated to point to the backward journal rather than the MIM in accordance with the backward journal move list, then the forward journal move list is processed to update the MIM and create space in the forward journal as before.

The VBMM and VBMJ data structures described here need not be constructed from simply binary trees. Multi-key, multi-pointer tree structures (such as B-trees or B+-trees) may be used instead, for greater retrieval efficiency. **Figure 19** depicts an exemplar multi-way VBMJ tree data structure (1900) for representing a mapping from logical storage device addresses to physical journal/snapshot addresses. Each of the nodes in tree 1900 represents a contiguous range of logical addresses, to be sure, but the data within the range need not be stored contiguously in the snapshots/journal entries. Instead, multiple pointers are provided from each node to represent each of a number of subranges making up the contiguous range of addresses. Thus, while node 1904 represents logical addresses from 40 to 860, there are pointers (1906, 1908, 1909, and 1911) pointing to physical representations of the subranges 40-66, 67-79, 80-300, and 301-859, respectively. Essentially, VBMJ 1900 is a VBMJ as described in **Figure 6** (VBMJ 602), but

with the physical extent lists incorporated into the tree nodes themselves, rather than as leaves of binary tree nodes. In this way, node accesses can be reduced, since although each node represents a contiguous range of
5 addresses, several different subranges may be identified by examining only one node. Using a multi-way tree, such as VBMJ 1900, reduces the number of memory or disk accesses used to retrieve nodes in the tree, and thus enhances the performance of the logical address mapping
10 function.

Figure 20 is a flowchart representation of a process of generating storage replicas in accordance with a preferred embodiment of the present invention. One should note that the steps depicted in **Figure 20**,
15 although they are executed in sequence with respect to a single journaled write command, in a preferred embodiment they will actually be performed in parallel on multiple items of data. For example, write commands will continue to be written to the journal while older write commands
20 are added to the VRMO. Thus, the preferred execution model is a "pipelined" or "assembly line" approach, where each step is performed simultaneously, but with respect to different items of data. First, an atomic write instruction is received from a controlling computer
25 system (step 2000). The write instruction is written in a journal and a VRMO (virtual recovery mapping object) is generated (step 2002). Certain conditions can cause a forward journal movelist to be constructed, namely the lack of space for additional journal entries or the
30 passage of a specified length of time (step 2004). If

Docket No. 2001-054-SFT

the criteria have not been met, the process cycles to step 2000. If so, however, the write instructions from the journal are combined to make a snapshot, which is stored in the collection of snapshots (step 2006). Next, 5 the "mirror in the middle" (MIM) is updated to match the snapshot (step 2008). If a criterion for recording to removable media has been met (step 2010), then image and difference "tapes" (or disks, etc.) may be generated from the snapshots (step 2012). In either case, the process 10 cycles again to step 2000. One should note that although steps 2010 and 2012 are shown here as part of a sequence of steps, steps 2010 and 2012 may, in fact, be performed asynchronously with respect to the generation of snapshots (i.e., at any time, not just following snapshot 15 generation) or not at all.

One of ordinary skill in the art will recognize that a suitable control computer program may be utilized by a user or administrator to set the criteria for when MIM updates or tape-transfer events will occur. These events 20 may be set to occur at a certain time of day or after a certain time-period has elapsed, or they may be set to coincide with other events, such synchronization of a database management system, for instance. One of ordinary skill in the art will recognize that the 25 scheduling of synchronization and/or tape-transfer events may be performed in any manner desired without departing from the scope and spirit of the invention.

Figures 21-23 illustrate particular applications for a data management appliance in accordance with a 30 preferred embodiment of the present invention. Having a

data management appliance with the ability to retrieve mirrored versions of a storage device from the past makes it possible for a monitor process to monitor for a troublesome change in condition of the data stored on the storage device and for the corrupted data to be restored to its latest correct state.

For example, **Figure 21** depicts a process of monitoring a database for violation of consistency constraints (such as values falling out of specified ranges or spurious duplicate or ambiguous data, for example). An application server (2100) makes use of a database stored on primary disk 2102. Data management appliance 2104 stores virtual mirrors 2106 of primary disk 2102 over time. A certification server 2108 can mount data management appliance 2104 and examine each individual virtual mirror (such as virtual mirror 2110) for correctness. If a problem is located, primary disk 2102 can be restored with the latest correct virtual mirror stored by data management appliance 2104. In an alternative embodiment, certification server 2108 may simply be replaced by software operating on protected application server 2100 or data management appliance 2104.

Similarly, **Figure 22** depicts a system that monitors for viruses. Application server 2200 makes use of a filesystem stored on primary disk 2202. Data management appliance 2204 stores virtual mirrors 2206 of primary disk 2202 over time. Virus scanner software 2208 can scan each individual virtual mirror (such as virtual mirror 2210) for viruses (or every other mirror, or every

Docket No. 2001-054-SFT

third, etc.). If a problem is located, primary disk 2202 can be restored with the latest uninfected virtual mirror stored by data management appliance 2204.

Figure 23 is a flowchart representation of a process of monitoring for troublesome changes in data backed up by a data management appliance in accordance with a preferred embodiment of the present invention. First, if the monitoring takes place external to the appliance itself, the external device mounts the data management appliance to be able to access its data (step 2300). A virtual mirror on the appliance is checked to see if it conforms to specified constraints (e.g., to be virus free, to be a consistent database, to be error free, etc.) (step 2302). If the constraints are satisfied (step 2304:Yes), the next virtual mirror in chronological order is examined (step 2306). If not (step 2304:No), then the mirror chronologically previous to the currently examined mirror is examined to see if it conforms to the constraints (step 2308). If it does not (step 2310:No), then the next previous mirror is examined (step 2312). If does (step 2310:Yes), then the uncorrupted data in the mirror is restored to the primary storage device (step 2314).

One of ordinary skill in the art will recognize that a number of variations on present invention may be achieved without departing from the scope and spirit of the invention herein disclosed. For example, while the preceding figures described a single data management appliance used in conjunction with a single primary storage device and a single computer system, in fact, the

Docket No. 2001-054-SFT

present invention may be utilized in a scaled fashion, with multiple appliances, multiple primary storage devices, and/or multiple computer systems being connected together in a storage network. **Figure 24**, for instance, depicts a single data management appliance (2400) attached to a storage network (2402) with multiple servers having attached primary storage devices (2404) being attached to storage network 2402. The primary storage devices may be mirrors of each other, or may possess different contents. All may shared the same data management appliance (2400).

Figure 25 depicts a single data management appliance console 2500 controlling multiple data management appliances 2504 which are managed as a single unit, without regard for the number of appliances actually included in the installation. This allows the capacity or activity level of primary storage to be increased without concurrently increasing the administrative effort required to keep the protection mechanism supplied by data management appliances 2504 operative. Tape library 2502 may be used to store image and difference tapes made from snapshots stored on data management appliances 2504. **Figure 26** shows a similar system wherein data management appliances 2600 share common pooled random access storage (2604).

It is important to note that while the present invention has been described in the context of a fully functioning data processing system, those of ordinary skill in the art will appreciate that the processes of the present invention are capable of being distributed in

the form of a computer readable medium of instructions and a variety of forms and that the present invention applies equally regardless of the particular type of signal bearing media actually used to carry out the
5 distribution. Examples of computer readable media include recordable-type media such a floppy disc, a hard disk drive, a RAM, CD-ROMs, and transmission-type media such as digital and analog communications links.

The description of the present invention has been
10 presented for purposes of illustration and description, and is not intended to be exhaustive or limited to the invention in the form disclosed. Many modifications and variations will be apparent to those of ordinary skill in the art. The embodiment was chosen and described in
15 order to best explain the principles of the invention, the practical application, and to enable others of ordinary skill in the art to understand the invention for various embodiments with various modifications as are suited to the particular use contemplated.